# Tailflow
# An OpenFlow Controller Framework

Torbjörn Törnkvist
*22 March 2013*

# Tail-f Systems

- Founded 2005
  - HQ in Stockholm Sweden, with US sales

- Software Products:
  - ConfD – On-device Management Agent
  - NCS – Network Control System

- Customers, + 75 world-wide including:

# What you'll hopefully will learn:

- SDN - what does it mean?
- Openflow - what is it?
- Modern Network Management - the Tail-f way
- Service Chaining - a realistic use case

# Network management (according to Wikipedia)

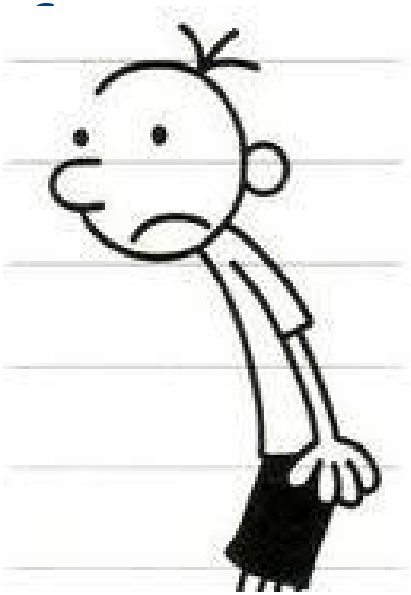
Refers to the:

- activities
- methods
- procedures
- tools

that pertain to the:

- operation
- administration
- maintenance

**Ugh...manual work...**
**...easy to get wrong...**
**...and very costly!**

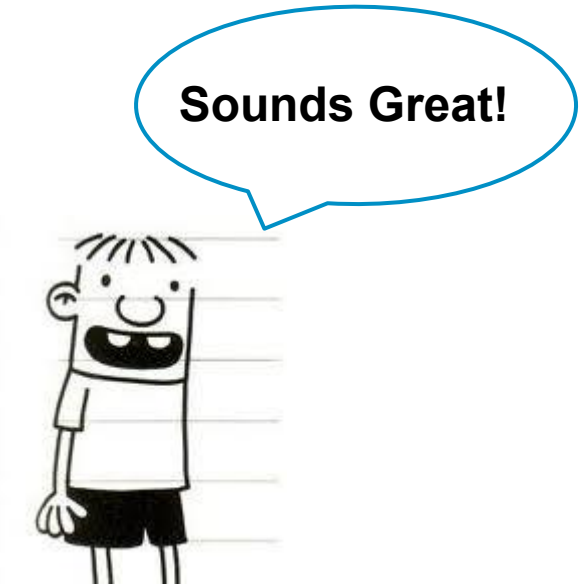


- A network management system (N[illegible]) monitor and administer a computer network networks.
- Common access methods include:
  - SNMP
  - Command-Line Interface (CLIs)
  - NETCONF

# Software Defined Networking (SDN)

- An approach to building computer networks that **separates** and **abstracts** elements of these systems.
- This makes it possible to apply modern software engineering techniques and practices.

- Reduces time to deployment
- **Reduces costs!**

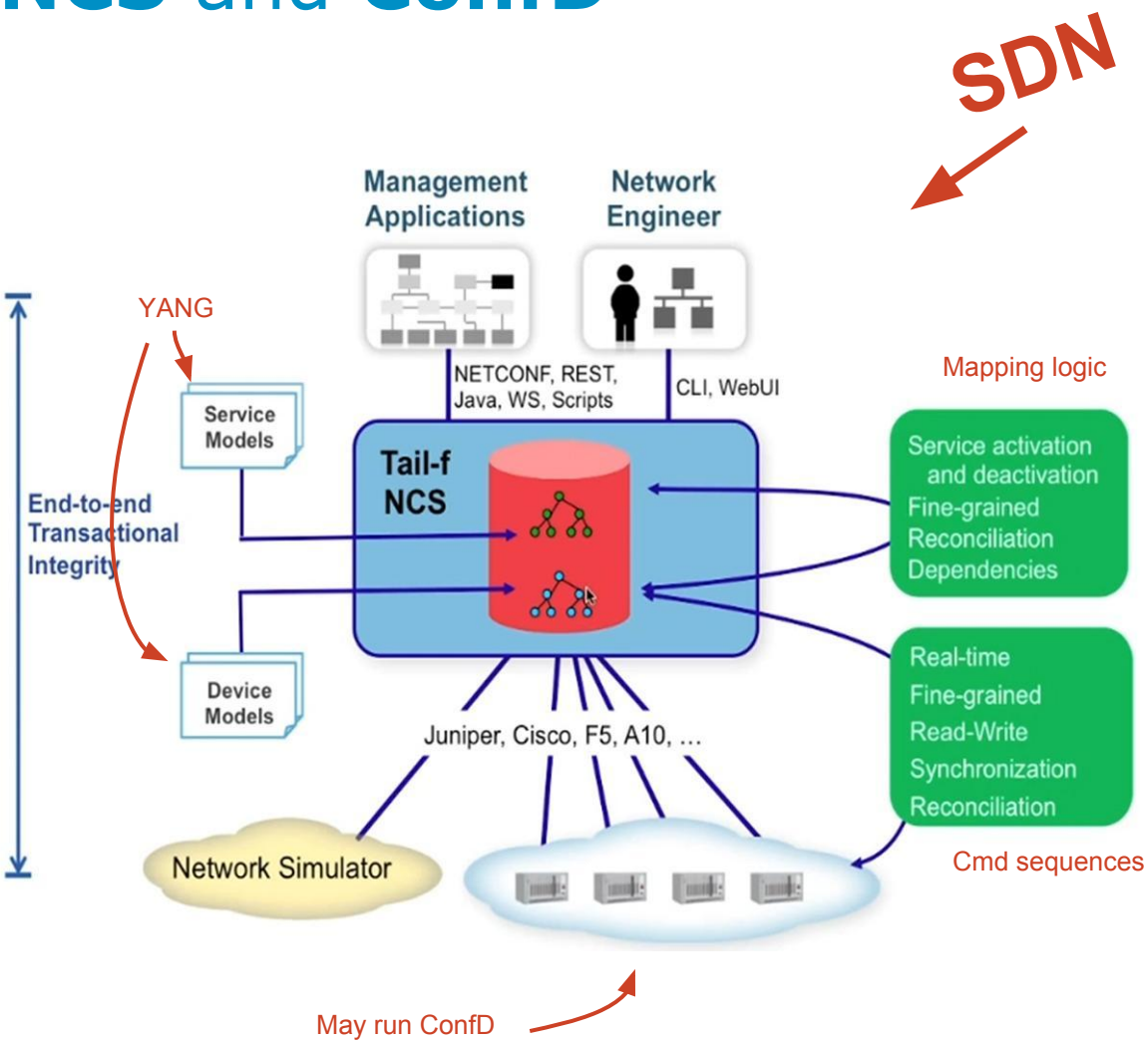


# **Tail-f** products: **NCS** and **ConfD**

**NCS** can control and configure a heterogenus set of network elements.

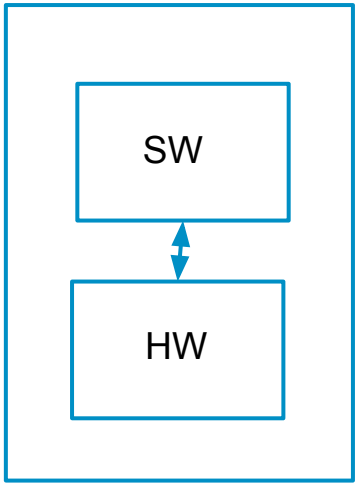
- *Models*
- *Datastructures*
- *Mapping logic*
- *Auto rendered interfaces*
- *Transactions*

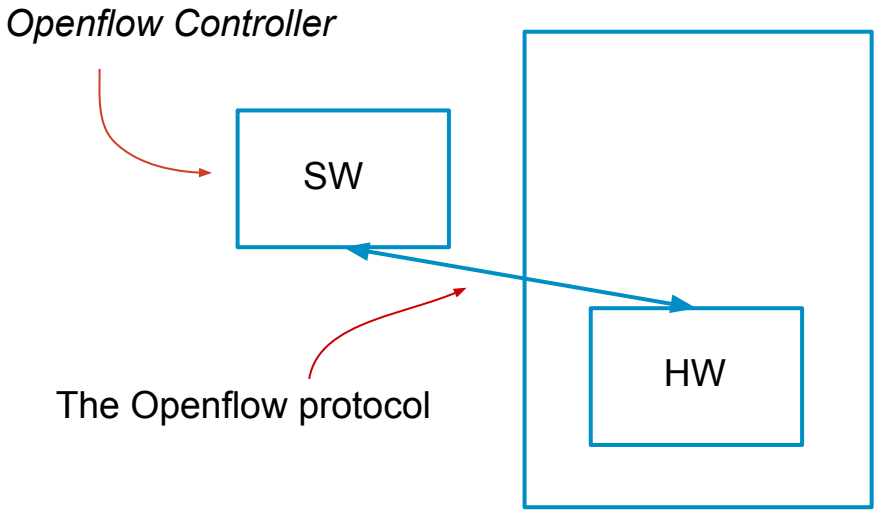
**ConfD** may run on the managed devices to provide CLI, NETCONF, SNMP... access.



# Openflow - what it is.

Traditional network element (according to my imagination)
(e.g a *switch/router/firewall*)

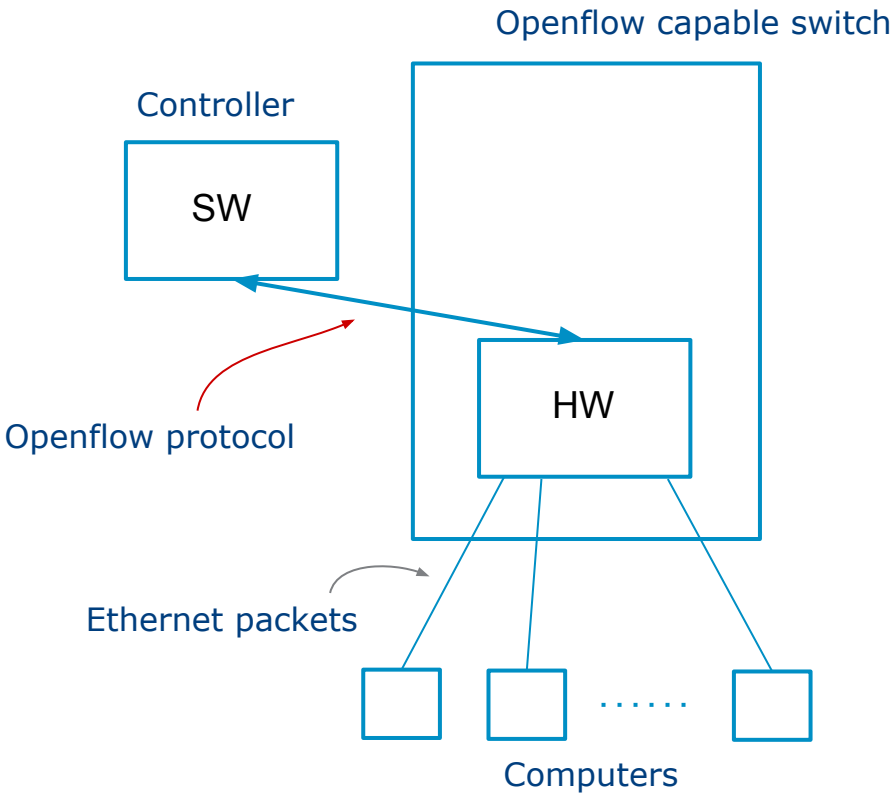



Openflow capable switch

# Openflow - the essence of it

- When a packet enters the HW, it looks into its **flow table**, to see what to do with it.
- Packet header values are **matched** against the flow table entries.
- A matching entry renders corresponding **actions** to be applied to the packet.



# Openflow - matching

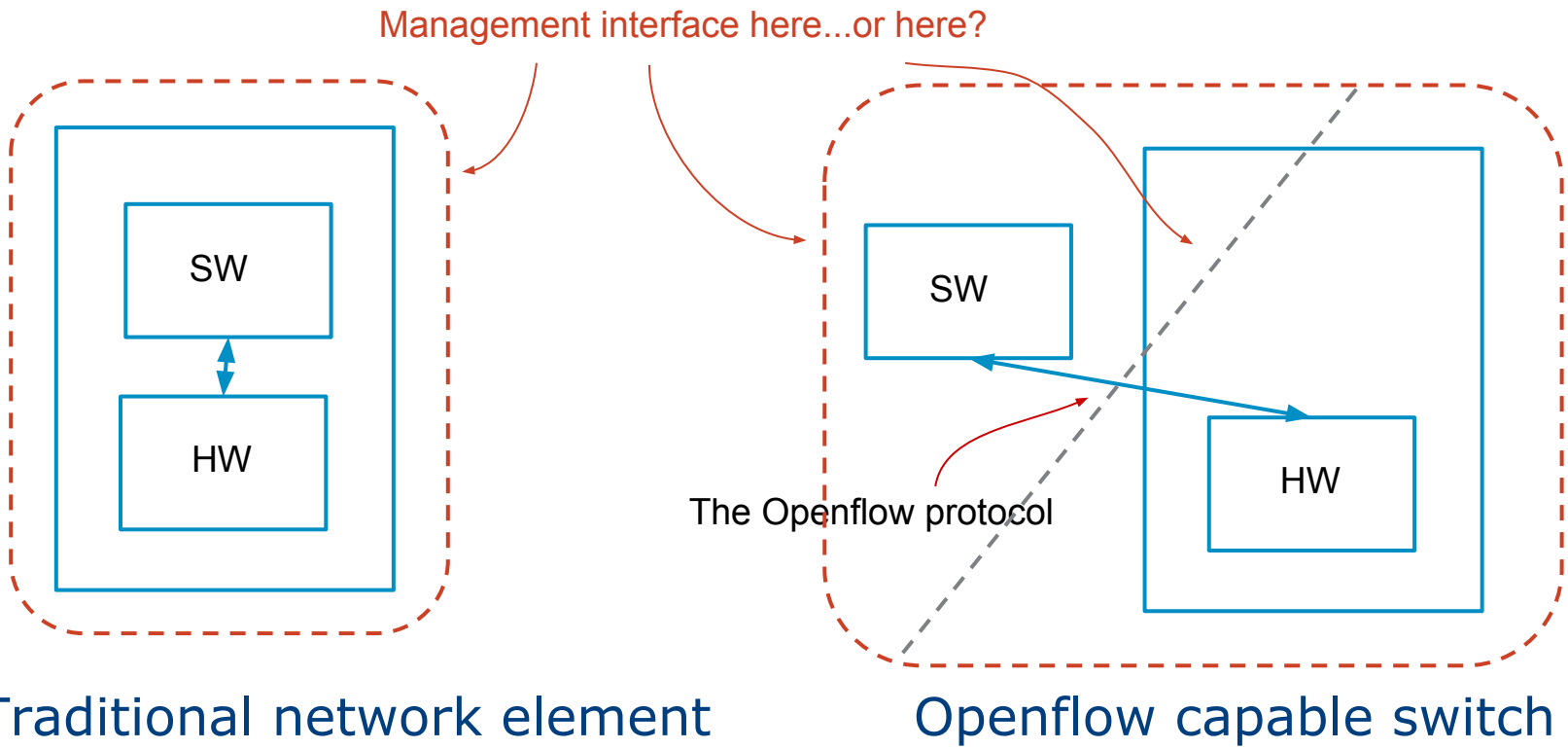
- If no **matching** entry is found in the flow table, then send the packet to the **controller** (SW) for some decision making.


flow table entry

- The SW tells the HW what to do now (and in the future by inserting a flow entry into the flow table of the HW).
- Coupled to a flow entry is a set of **actions**.
- Example of some actions:
  - Send out packet on <*port*>
  - Rewrite the <*ether source*> to some other address
  - Drop the packet (i.e no actions)

# Openflow device management

All kind of devices need to be managed



# SDN vs Openflow

- **Openflow is a component of SDN**

---

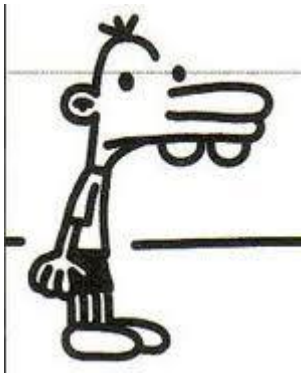
# Software Defined Networking (SDN)

- It's not really about programming the network.
- **It's about programming network services!**

# Part2: How to write an OF application?

**How can/should the SW be structured?**

**What about management?**



# The role of the controller

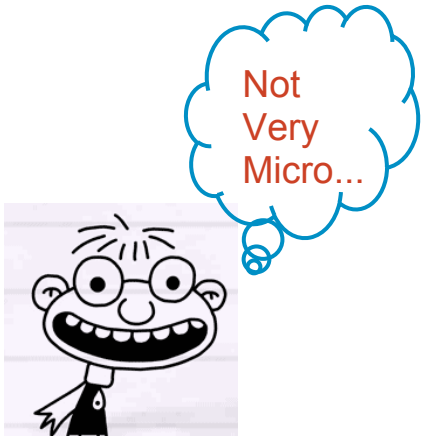
- From the Openflow controller point of view; an Openflow switch generates a number of events, for example:
  - *datapath-join* - when a switch connect to the controller
  - *packet-in* - when a packet is delivered from a switch
  - *flow-removed* - a flow (rule) was removed
    (e.g because of an expired timeout)

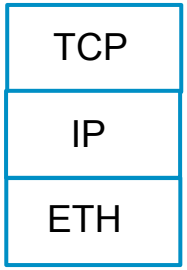
*For each event we want to apply some logic!*

# Sources of inspiration

- Functional programming (of course...)
- The micro-protocol idea (what?)

*By partition complex protocols into simple micro protocols, each of which is implemented by a protocol layer. Protocol layers can be stacked on top of each other in a variety of ways.*

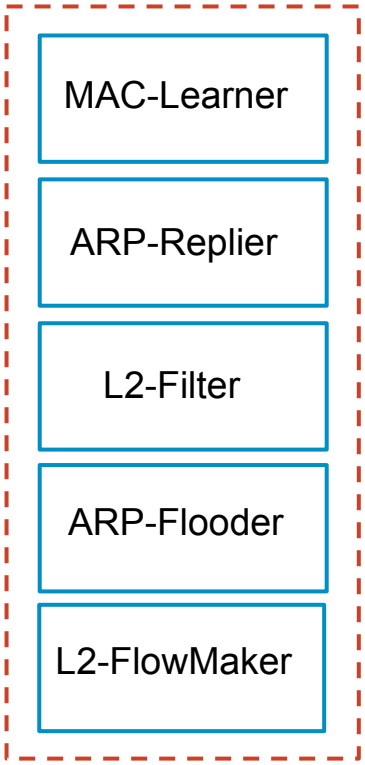



*Each layer encapsulates some minimal amount of logic in order to make it composable and easy to understand.*

# Somewhat more micro...

A stack of **flowlets**, forming a *virtual-L2-networks* application.




The *MAC-learner* snoop the incoming packets and store any new Mac+Port info in a DB

The *ARP-replier* can act as an ARP-proxy for a set of IP addresses.

The *L2-filter* will filter out which of the incoming packets that should be allowed further treatment of succeeding flowlets.

The *ARP-flooder* make sure that any incoming ARP request is flooded on all the switch ports.

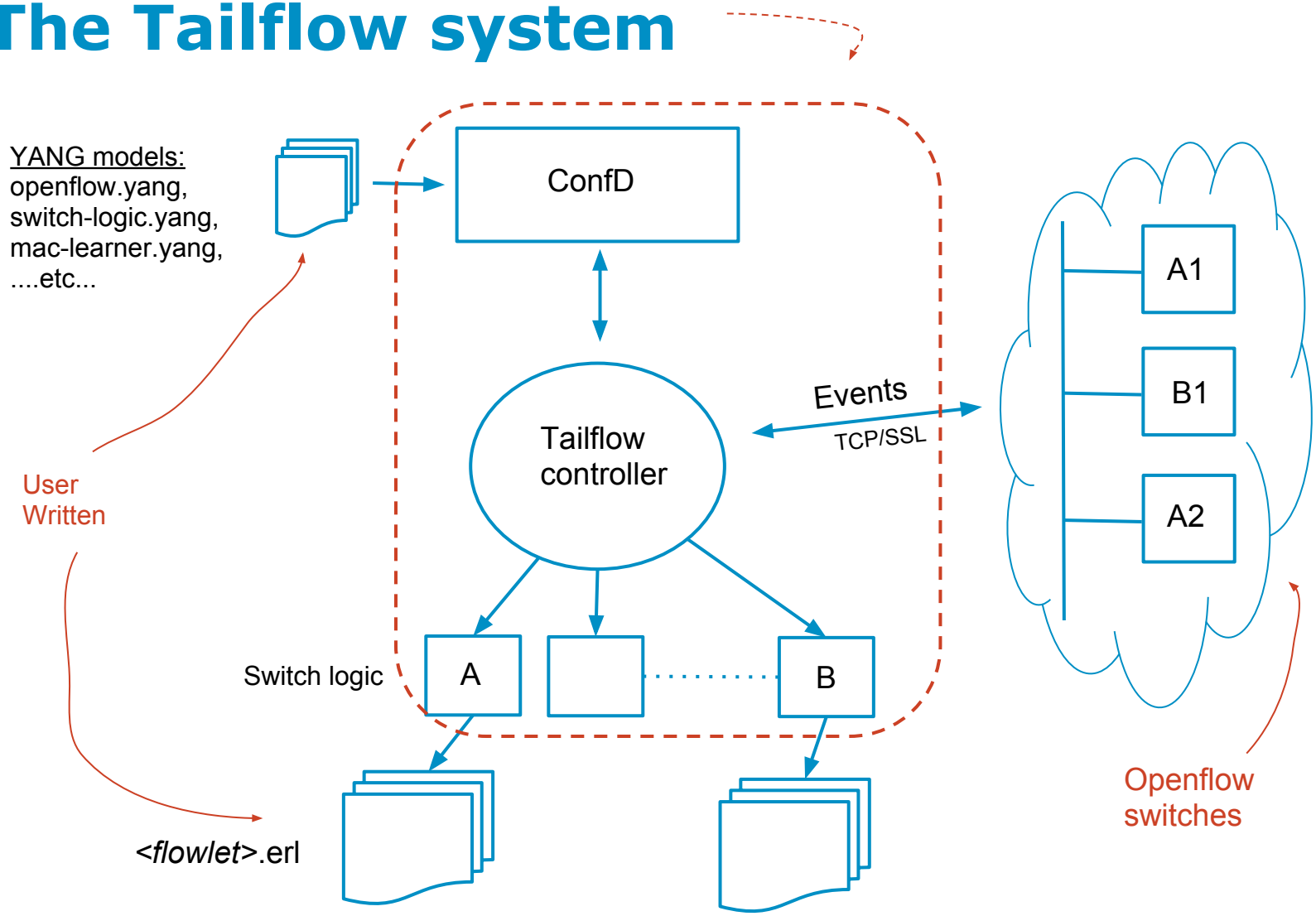
The *L2-flow-maker* creates an L2 flow for any incoming packet it sees.

# Tailflow key components

- **Flowlet ::= Erlang module + Yang module**
  - The configuration is described by Yang
  - Erlang modules are ordered in execution stacks
  - An Erlang module can return either of:
    - {break, LocalState}
    - {continue, LocalState, EventState}
    - {jump, LocalState, EventState, NewStack}

- **Switch-logic ::= Flowlet configs + Flowlet stacks**
  - Forms the complete software controlling the switch.
  - Can be applied to a particular switch (*datapath_id*)
  - Or to any switch

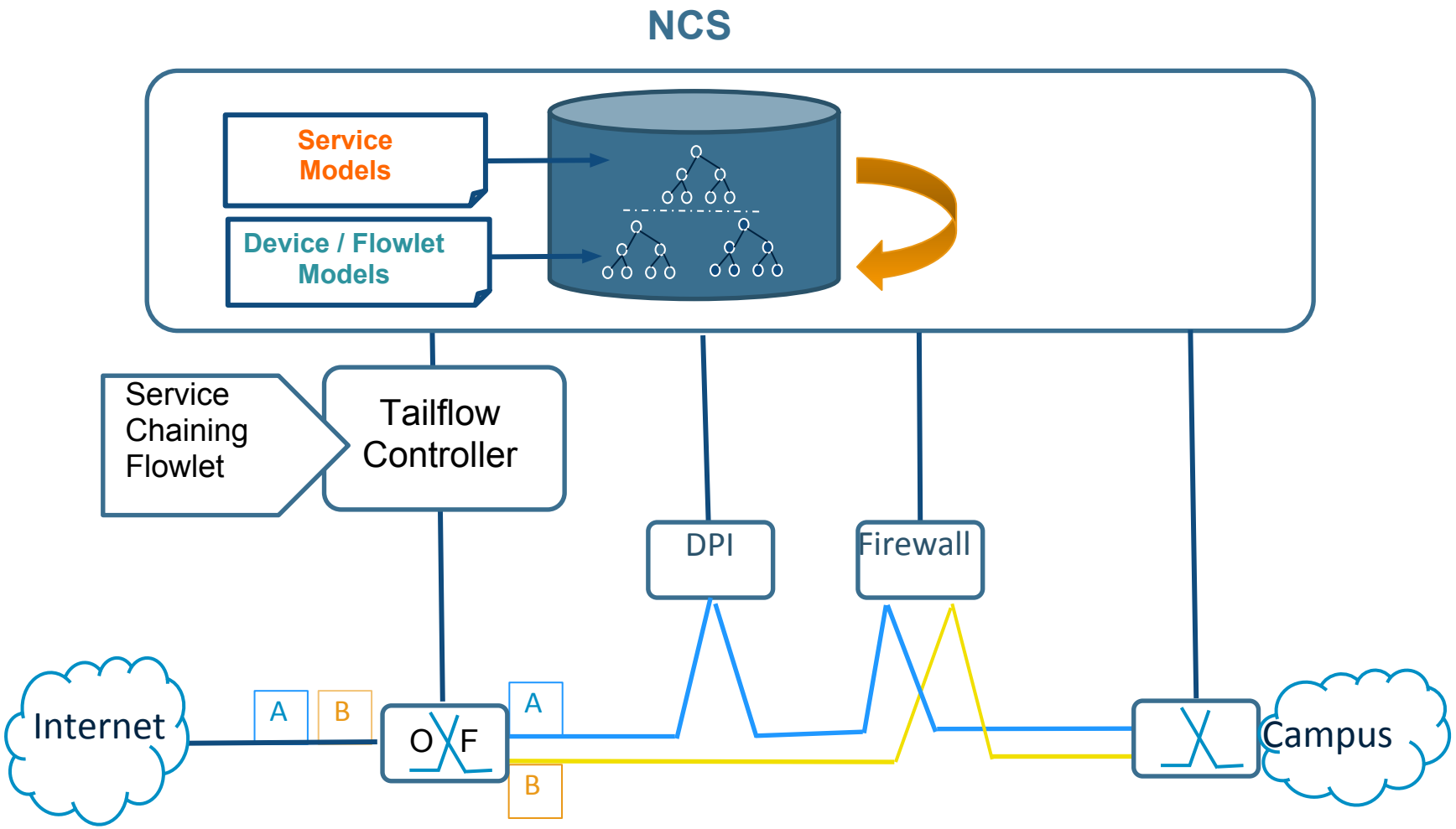
# The Tailflow system



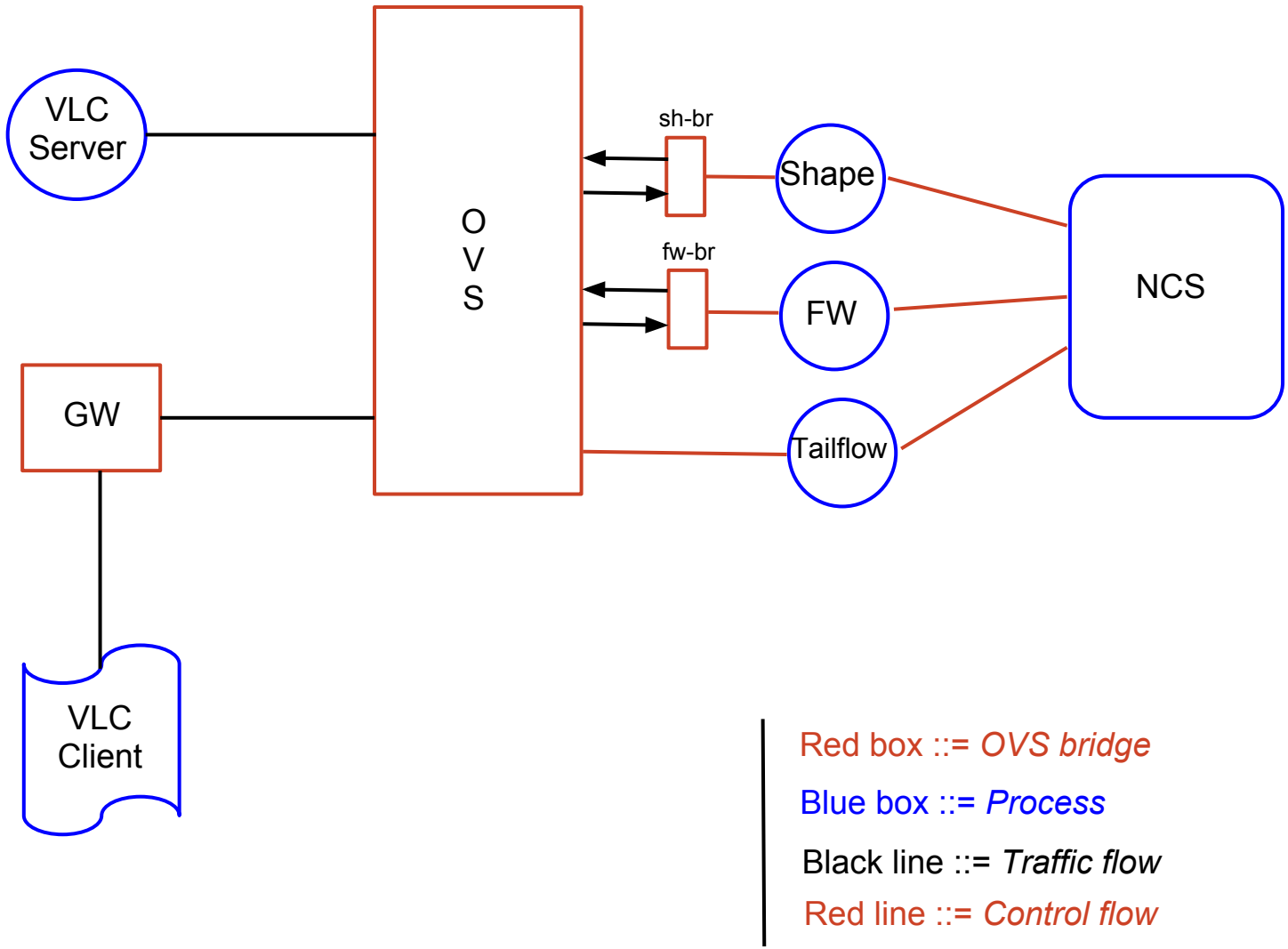
# Openflow usage - a realistic scenario

- Based on SrcNet, Vlan, etc... we want to thread packets through a chain of 'bump-in-the-wire devices.
- Each device implements some kind of service, e.g firewall, wlan-accelerator, deep packet inspector, etc...
- This is hard to do today with e.g policy routing, but easy with Openflow.

# SDN Use Case: Service Chaining



# Our demo setup:



# DEMO TIME!

Demo created by:

- Torbjörn Törnkvist (Tail-f)
- Claes Wikström (Tail-f)
- Luke Gorrie (snabb.co)
- Niklas Bystedt (Dataduktus)

# Thank you for listening!

# Questions?